CS 170 DISCUSSION 10

MAXIMUM FLOW

Raymond Chan
raychan3.github.io/cs170/fa17.html
UC Berkeley Fall 17

MAXIMUM FLOW

- Given a directed graph G = (V, E), send as many units of flow from source node *s* to sink node *t*.
- Edges have capacity constraints.
- When sending flow through some path, flow must be at most the capacity constraint of every edge.
- Number of units sent from *s* = number of units received by *t*.
- Applications: supply chain shipping, network flows, supply and demand shipping (need modification).

MAXIMUM FLOW

Max Flow Linear Program

$$\text{Maximize} \quad \text{size}(f) = \sum_{(s,u)\in E} f_{su}$$

$$\text{Subject to} \quad \sum_{(w,u)\in E} f_{wu} = \sum_{(u,v)\in E} f_{uv} \quad \forall u \in V \quad \text{Flow conservation consraints}$$

$$0 \leq f_e \leq c_e \quad \forall e \in E \quad \text{Capacity and nonnegativity constraints}$$

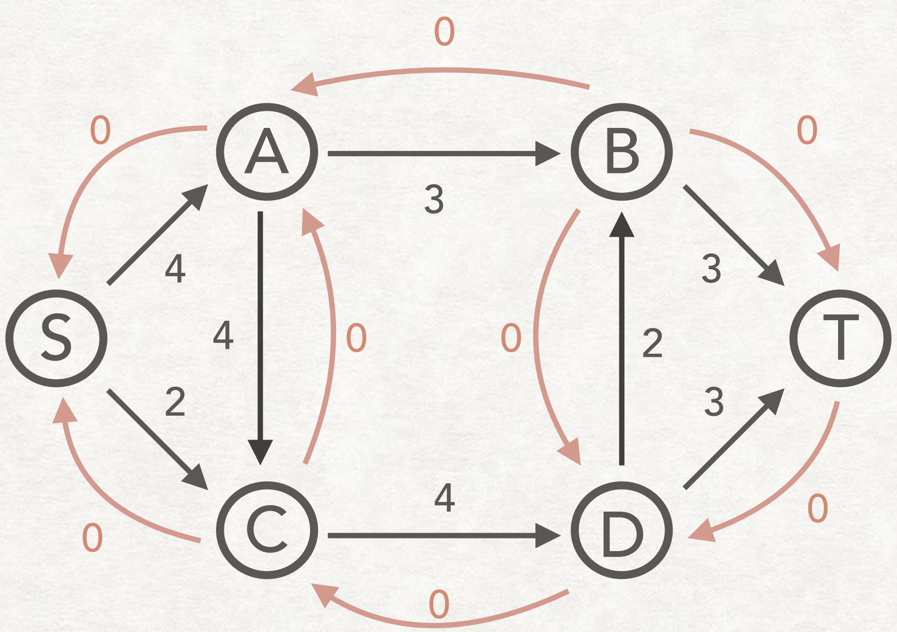
RESIDUAL GRAPH

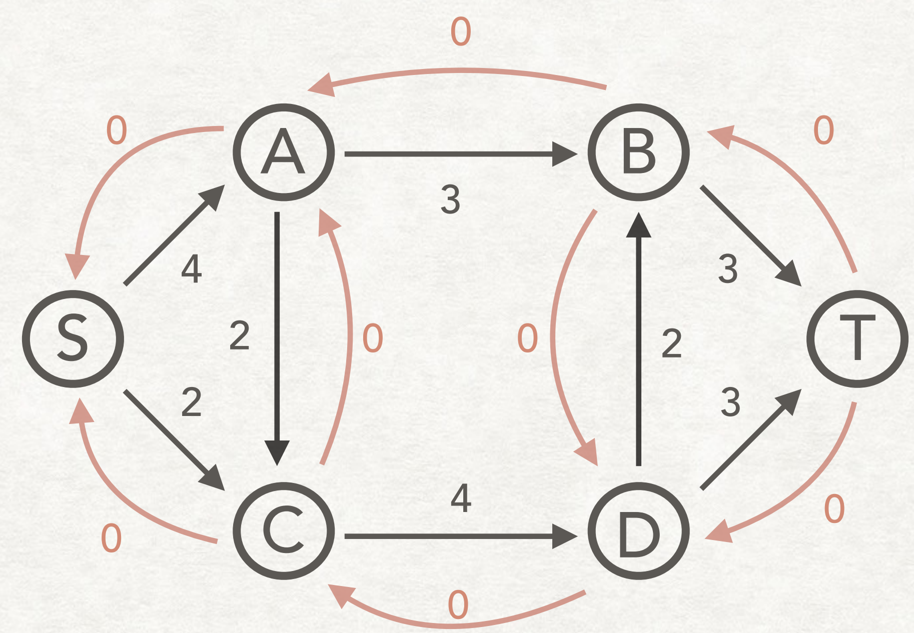
- When sending f_{uv} through edge (u, v), we create a residual back edge (v, u).
- Capacity of (u, v) = $c_{uv} - f_{uv}$
- Capacity of (v, u) = f_{uv}
- Keep sending flows from *s* to *t*.
- Once we have no path from *s* to *t*, we have the maximum flow.
- Edges can be removed if capacity reaches 0. This will ensure we won't have a path eventually.
- Can use residual edges later to go back on previous flow decisions.

MAX FLOW ALGORITHMS

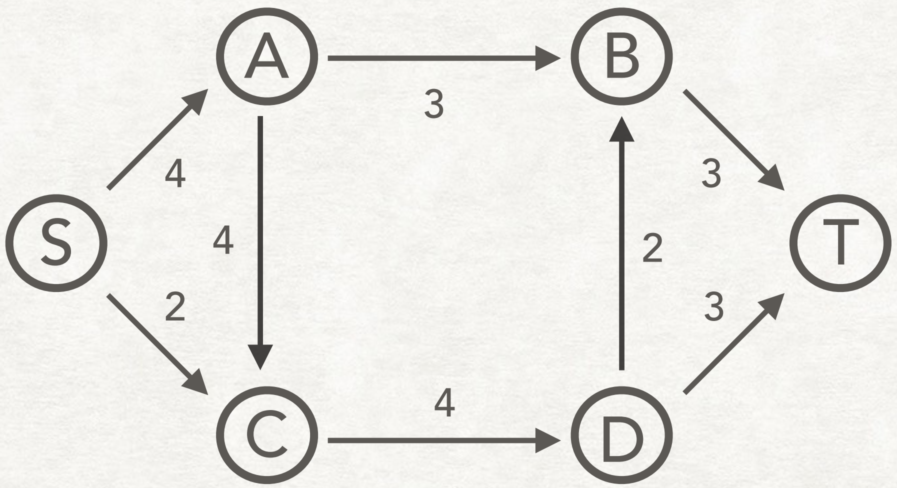
- Ford-Fulkerson algorithm.
  - In each iteration, send as many units of flow as possible and build residual graph.
  - Stop condition: no *s*-*t* path in residual graph.
- Edmond-Karp algorithm.
  - Find shortest *s*-*t* path via BFS in terms of edge hops.
  - Send as many units of flow as possible and build residual graph until no *s*-*t* path.
  - If capacities are integral, solution flows are integral, runtime is $O(|V||E|^2)$.

MAX FLOW RESIDUAL GRAPH EXAMPLE



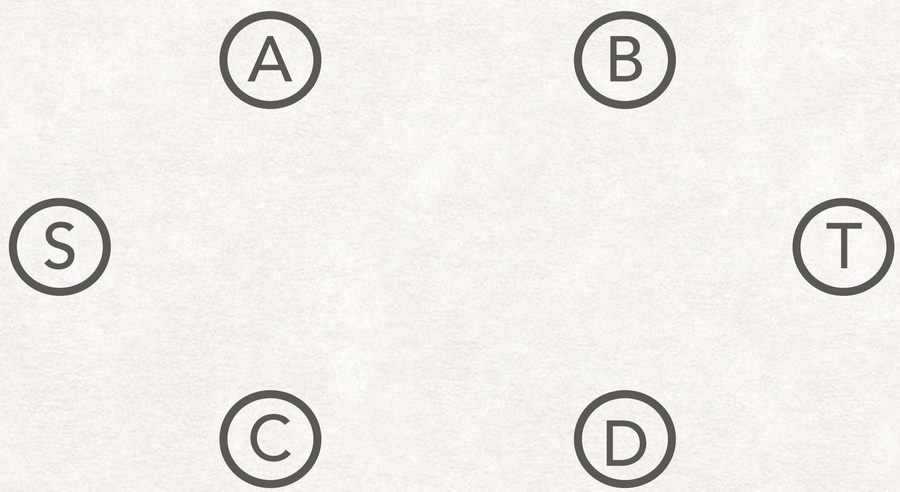


MAX FLOW RESIDUAL GRAPH EXAMPLE

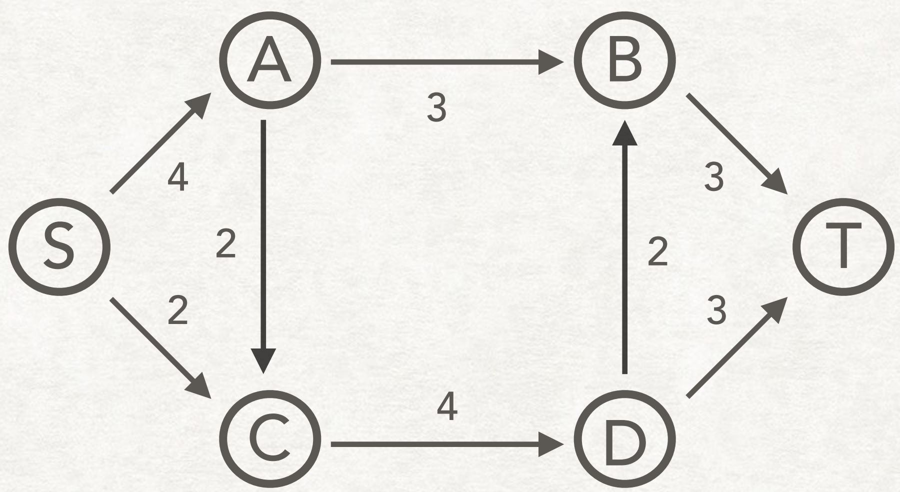


MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph

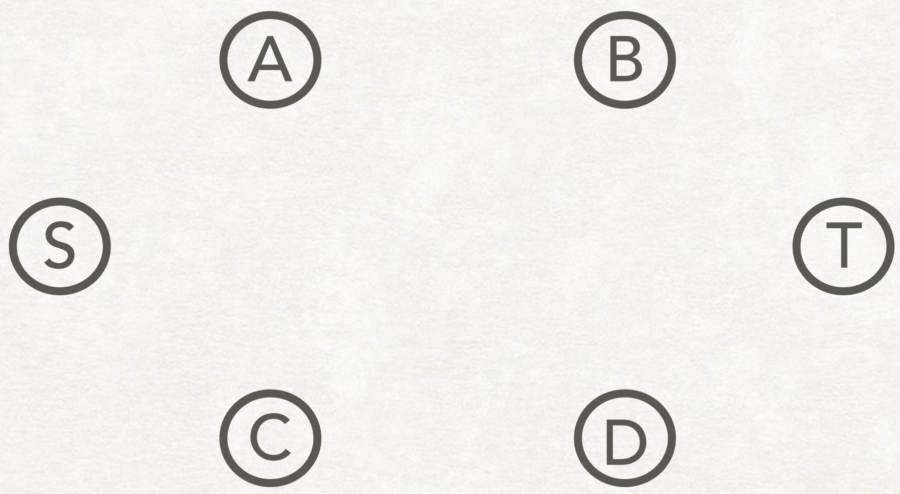


Residual graph

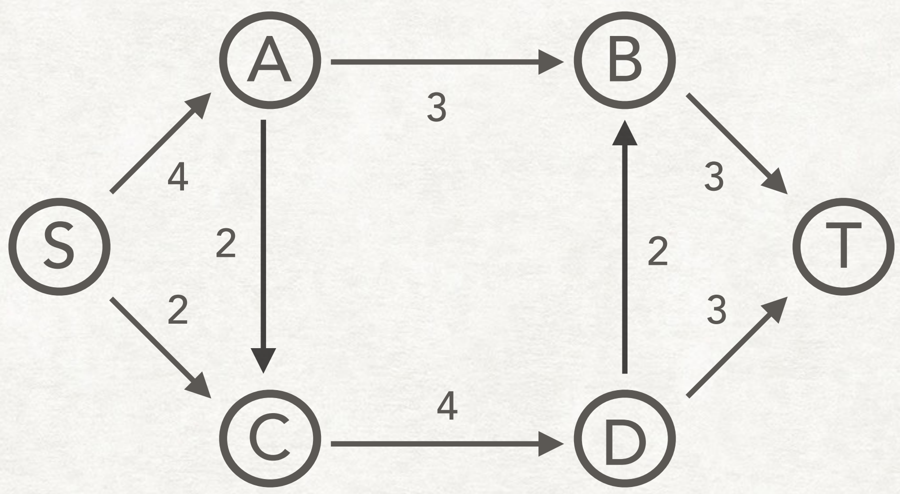


MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



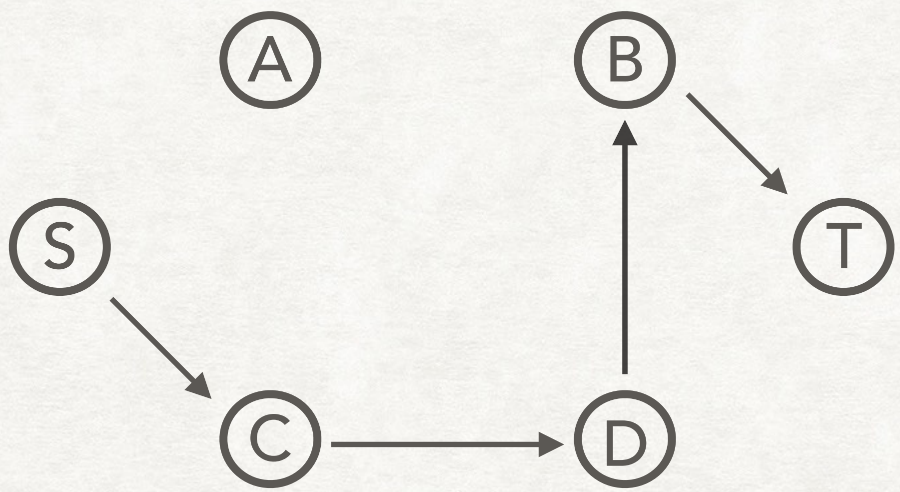
Residual graph



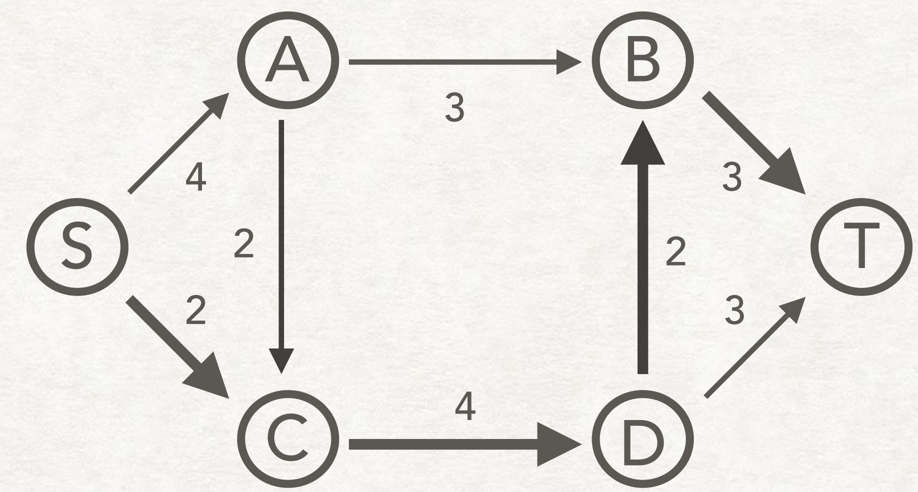
Suppose we send units through SCDBT first.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



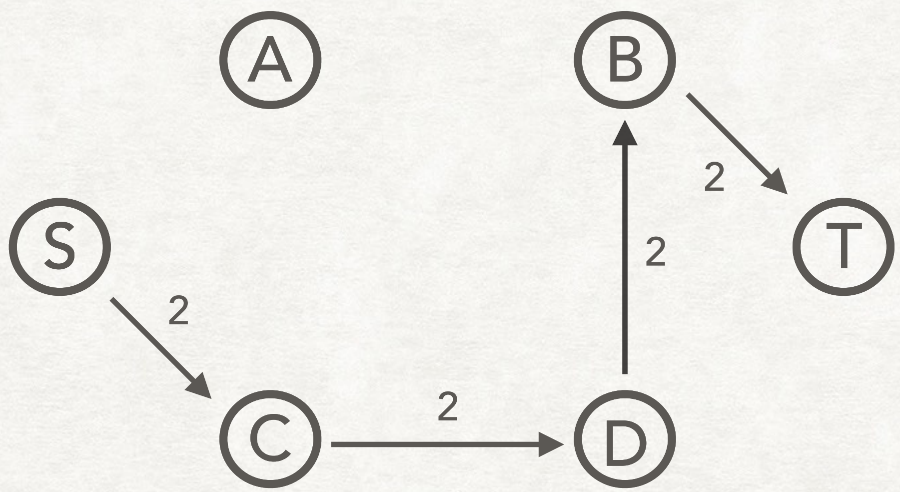
Residual graph



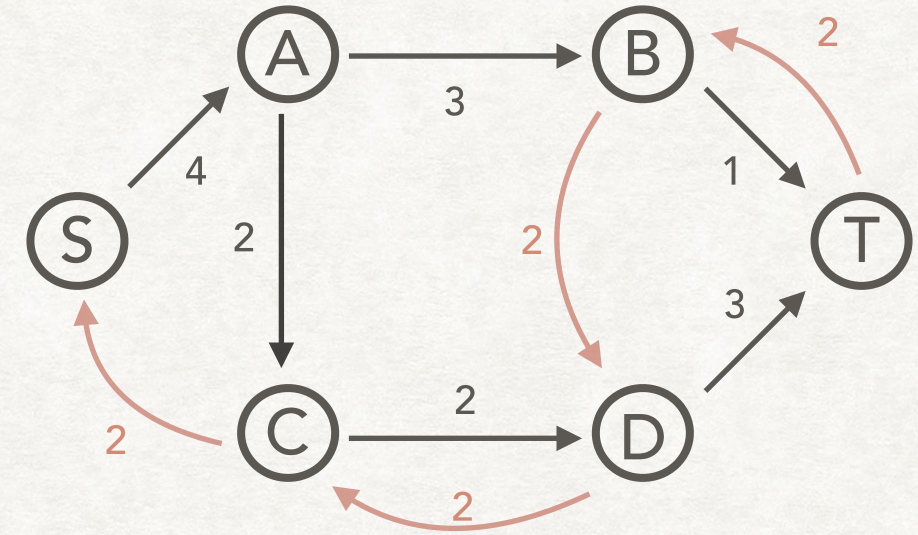
Send 2 units via SCDBT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



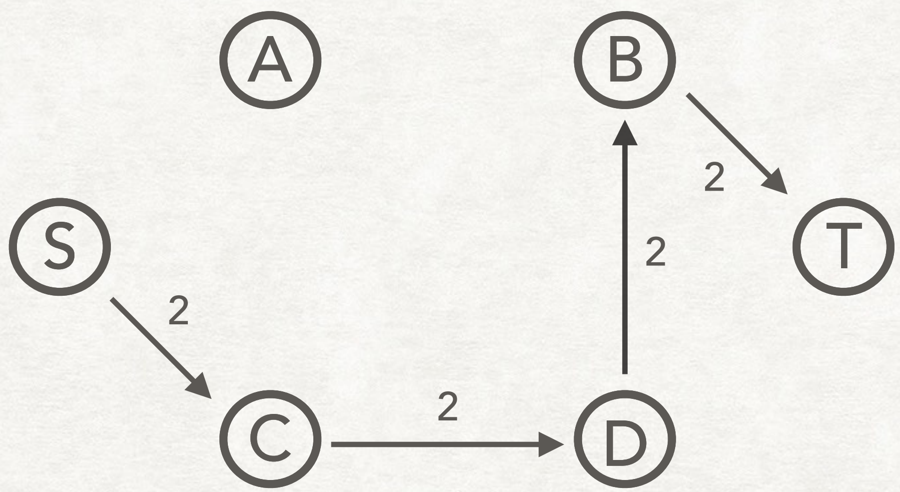
Residual graph



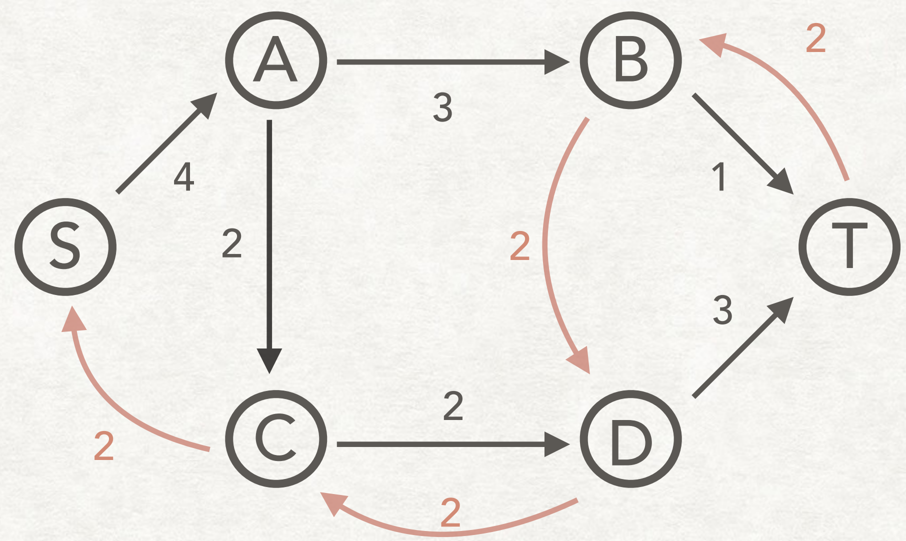
Send 2 units via SCDBT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



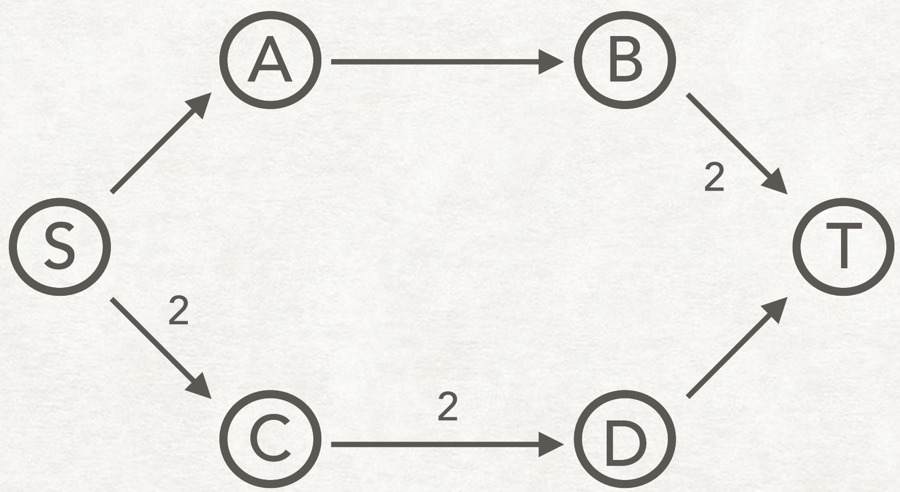
Residual graph



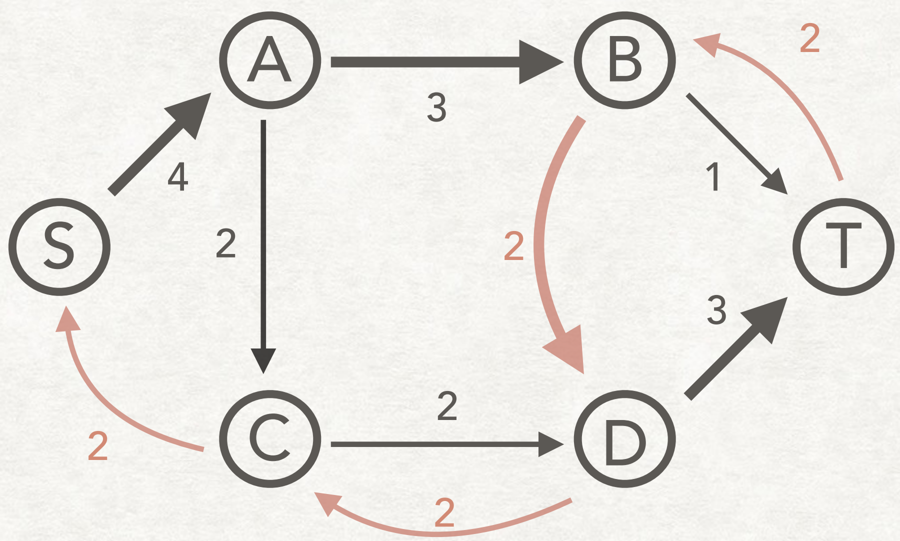
Send 2 units via SCDBT.
Send 2 units via SABDT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



Residual graph

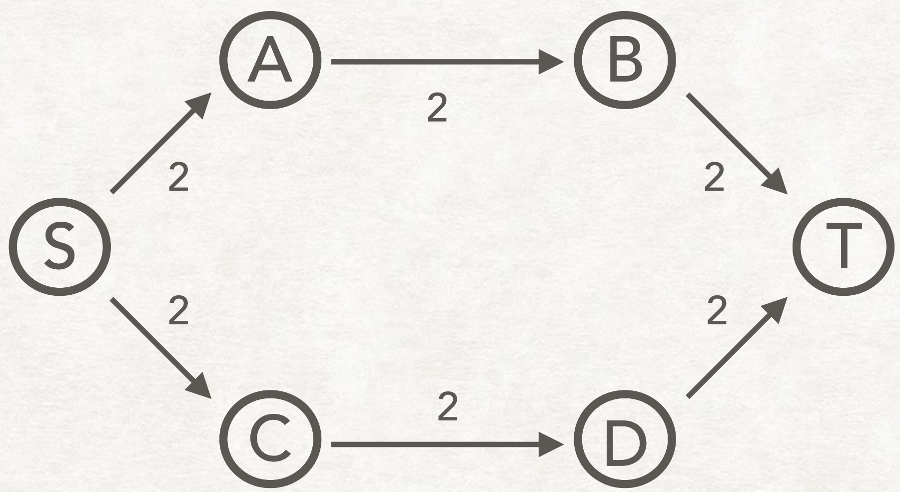


Send 2 units via SCDBT.

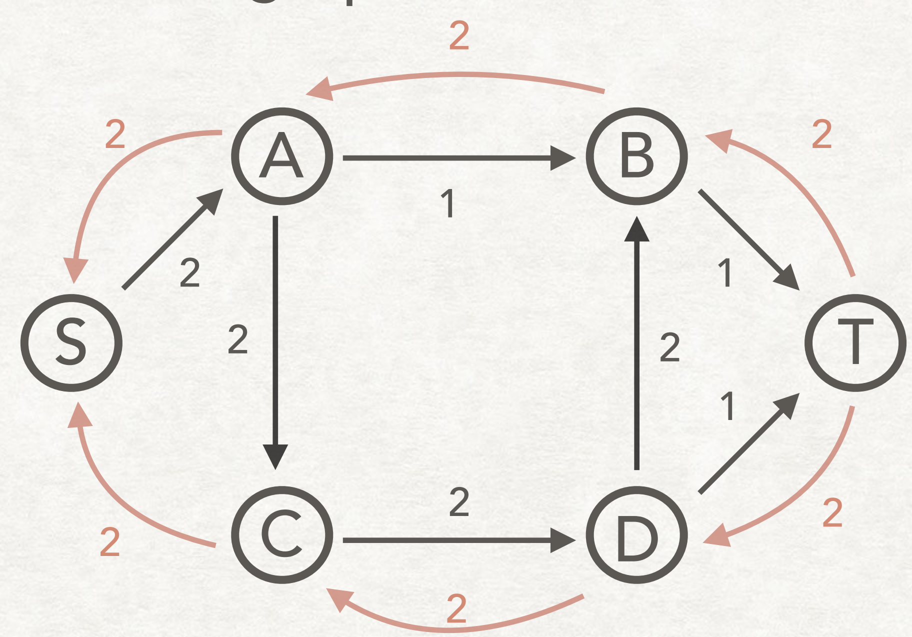
Send 2 units via SABDT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



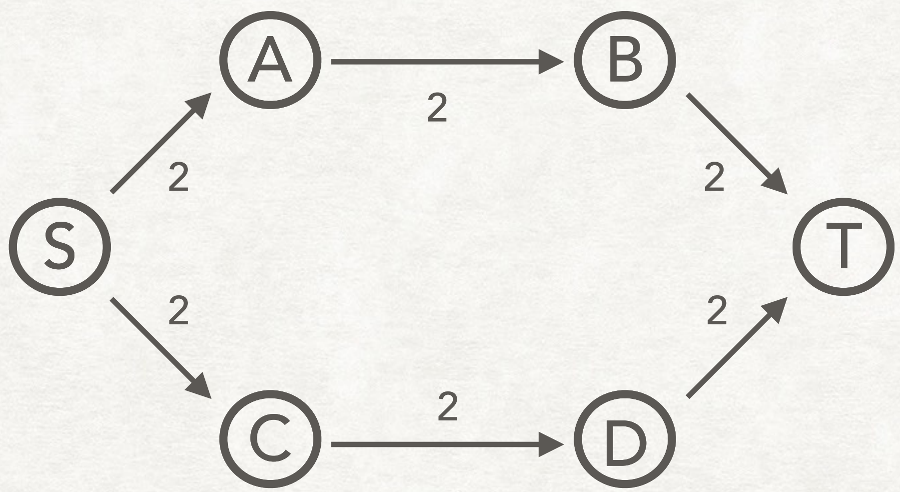
Residual graph



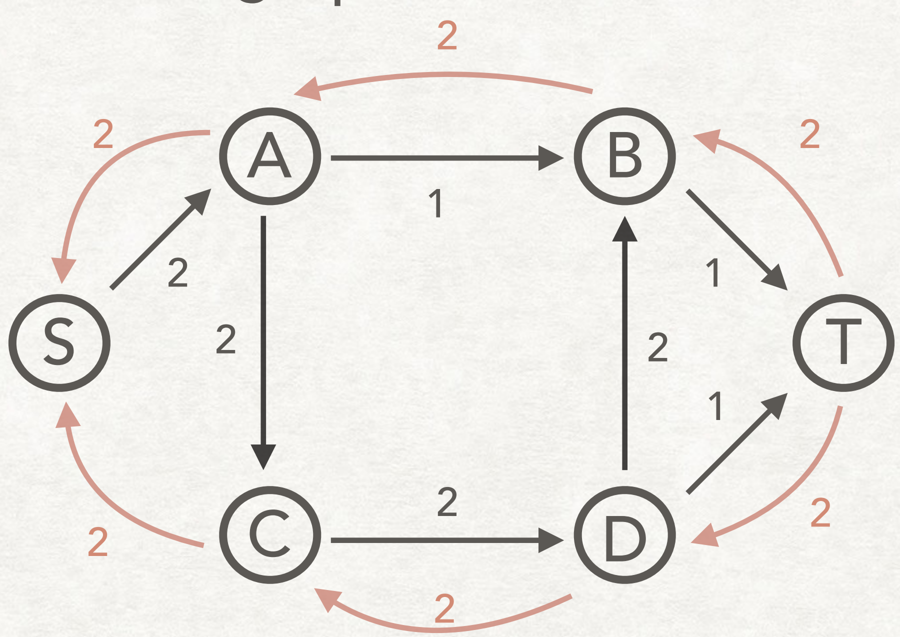
Send 2 units via SCDBT.
Send 2 units via SABDT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



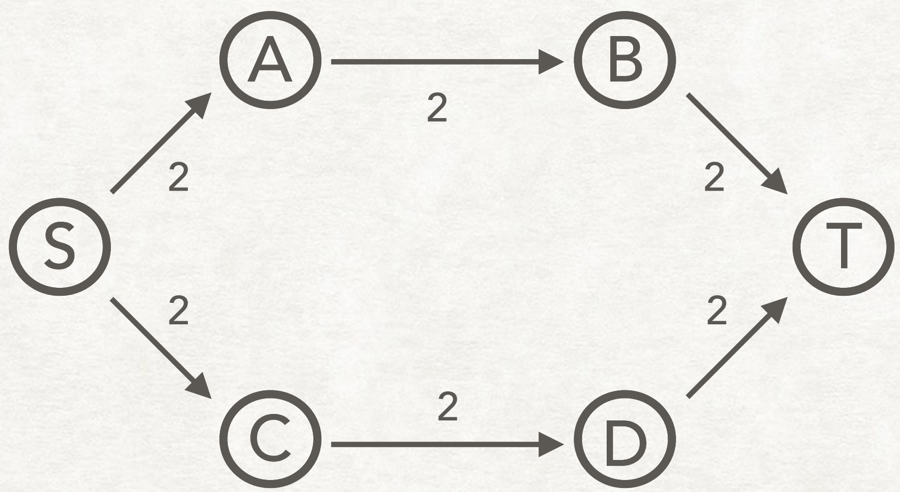
Residual graph



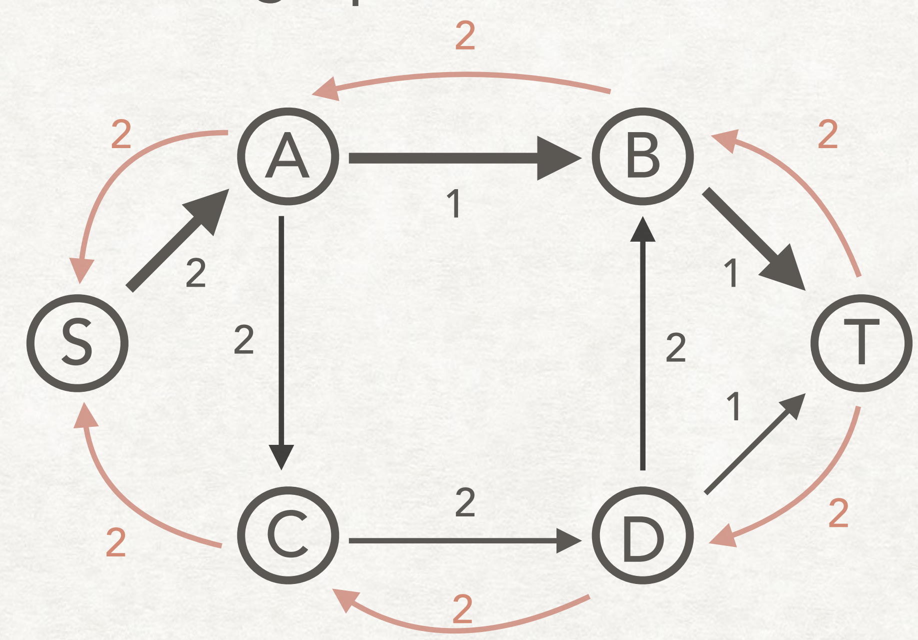
Send 2 units via SCDBT.
Send 2 units via SABDT.
Send 1 unit via SABT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



Residual graph



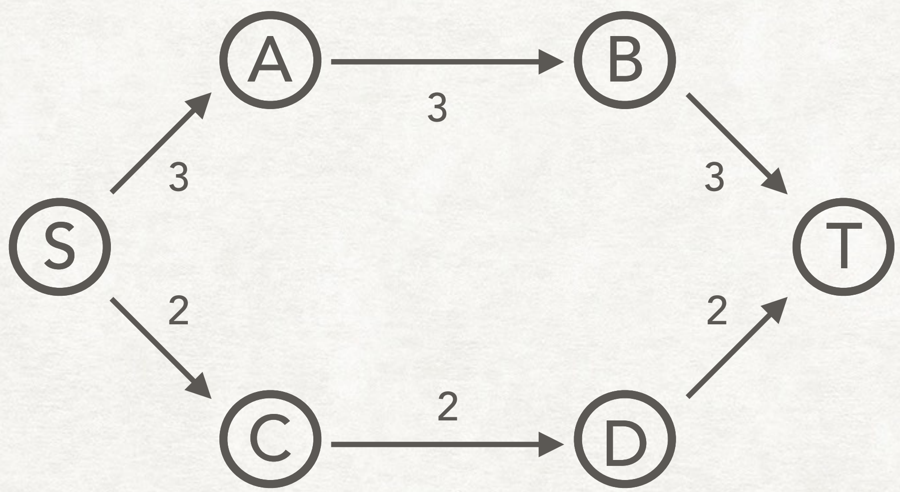
Send 2 units via SCDBT.

Send 2 units via SABDT.

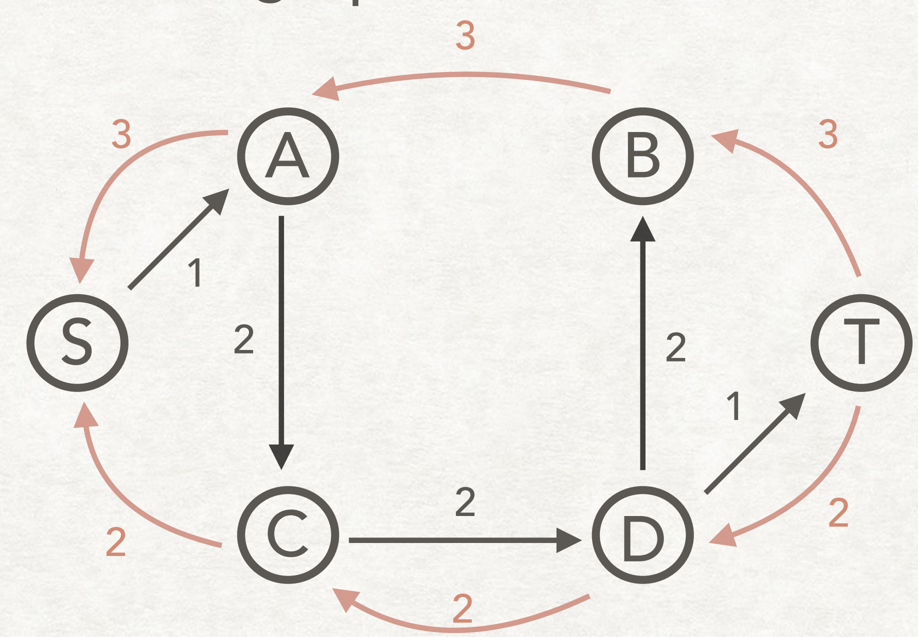
Send 1 unit via SABT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



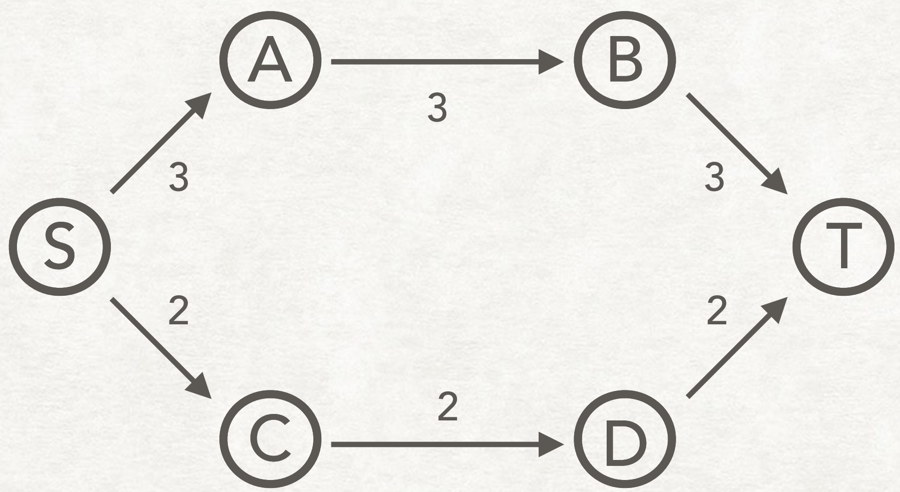
Residual graph



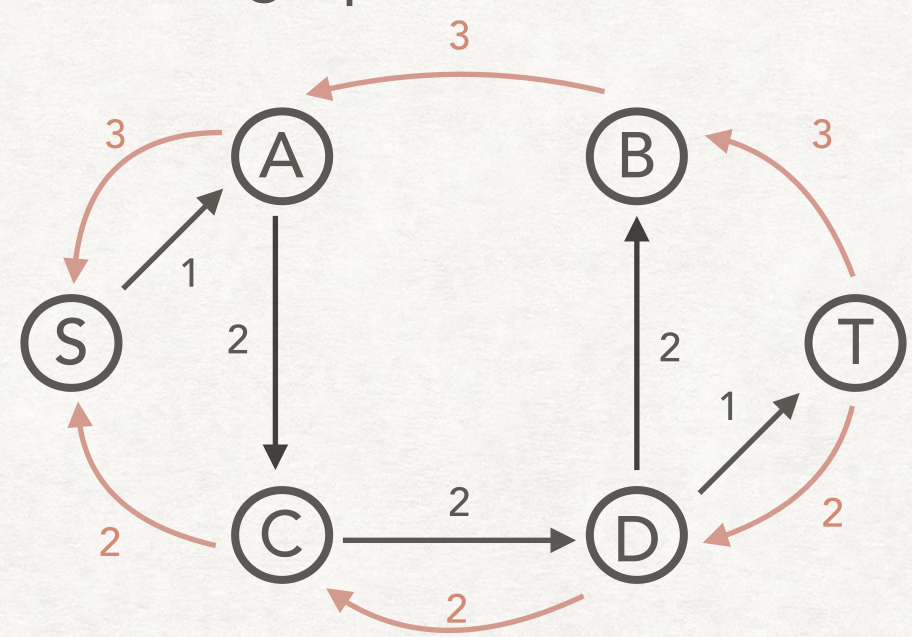
Send 2 units via SCDBT.
Send 2 units via SABDT.
Send 1 unit via SABT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



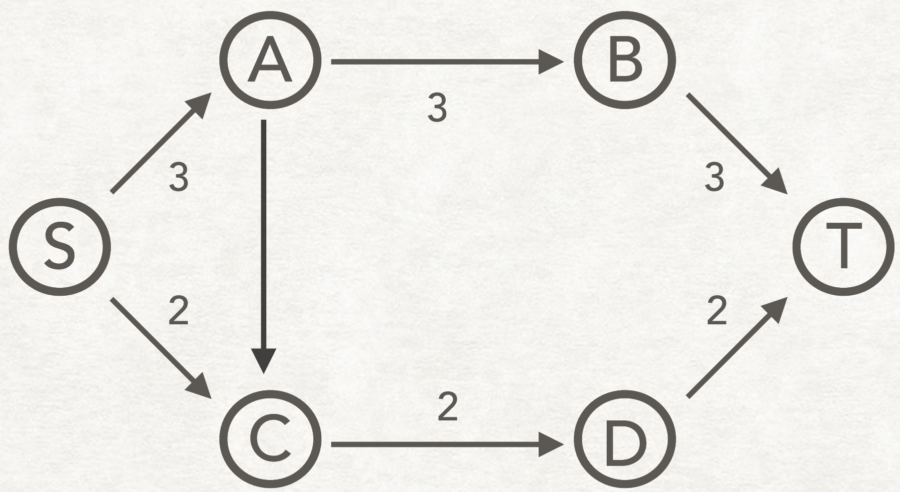
Residual graph



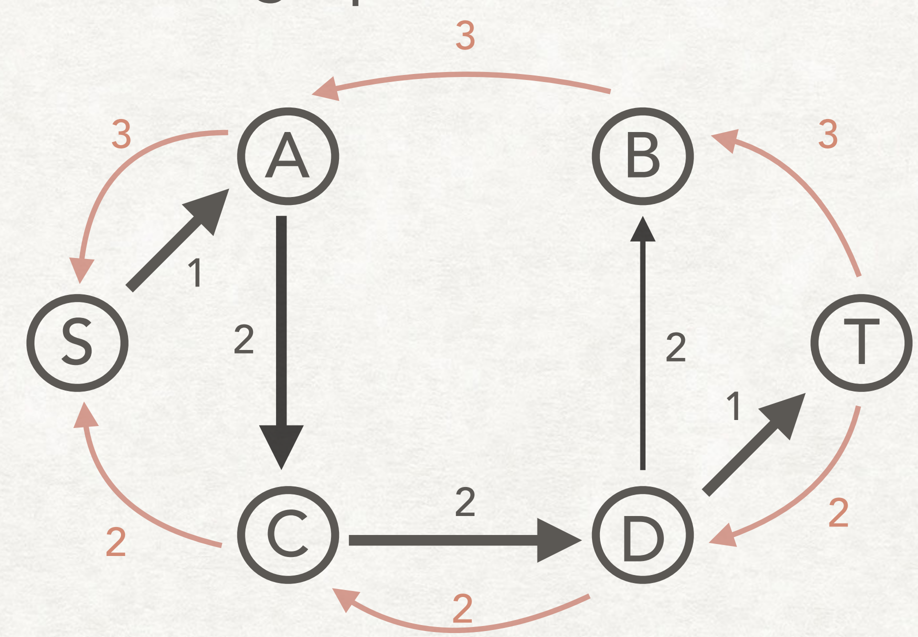
Send 2 units via SCDBT.
Send 2 units via SABDT.
Send 1 unit via SABT.
Send 1 unit via SACDT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



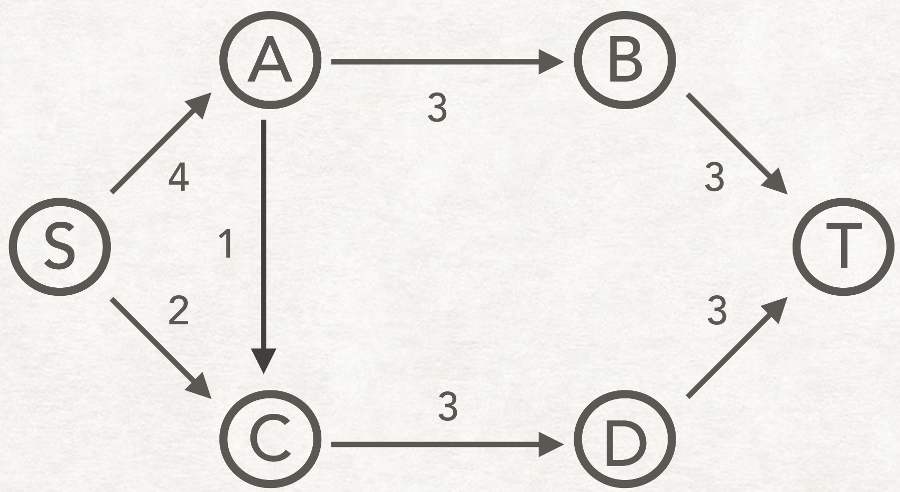
Residual graph



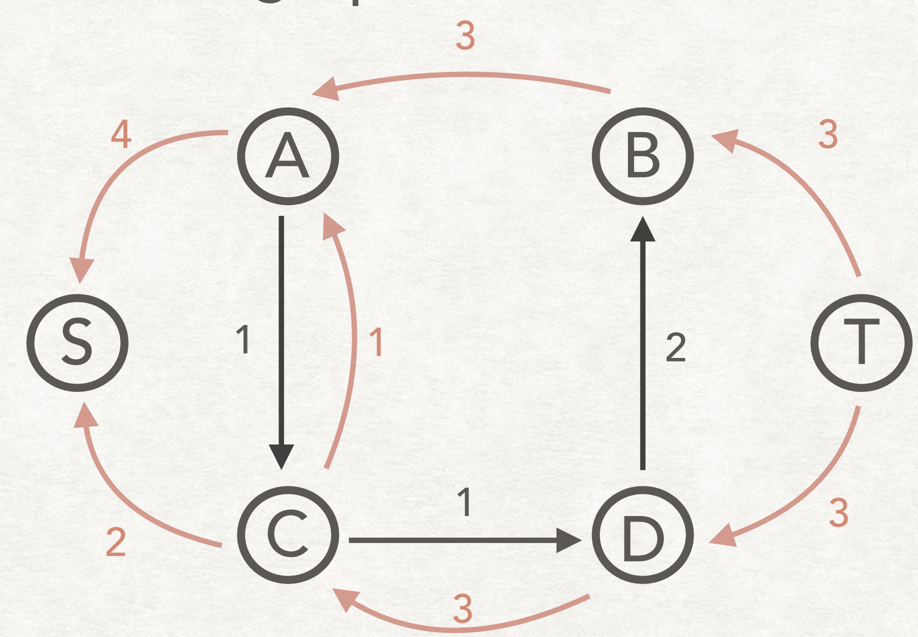
Send 2 units via SCDBT.
Send 2 units via SABDT.
Send 1 unit via SABT.
Send 1 unit via SACDT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



Residual graph



Send 2 units via SCDBT.

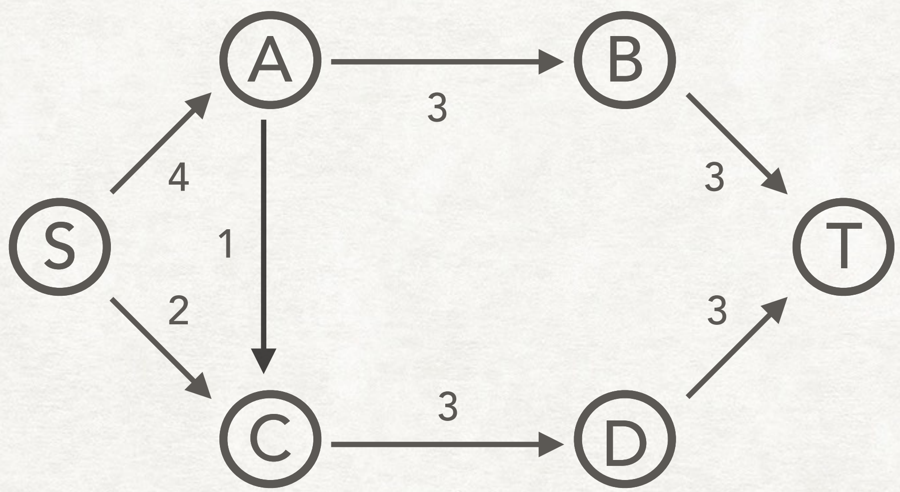
Send 2 units via SABDT.

Send 1 unit via SABT.

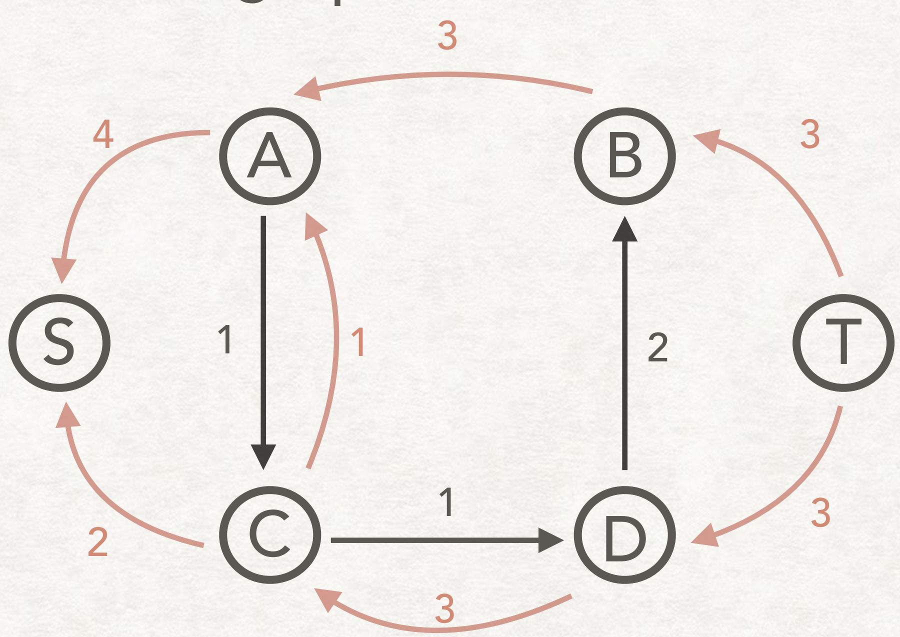
Send 1 unit via SACDT.

MAX FLOW RESIDUAL GRAPH EXAMPLE

Flow graph



Residual graph



Send 2 units via SCDBT.
Send 2 units via SABDT.
Send 1 unit via SABT.
Send 1 unit via SACDT.

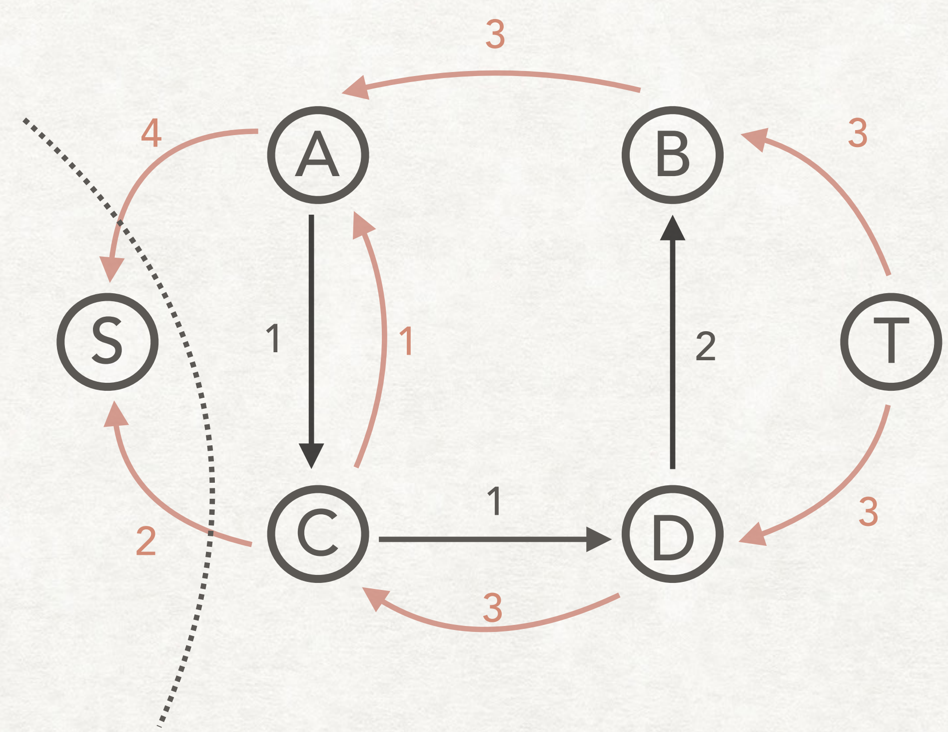
No more *s-t* path. Maximum flow is 6.
After computing max flow, output edges with their respective flows (see flow graph).

MINIMUM CUT

- A cut partitions the vertices into two disjoint sets L and R.
- The capacity of a cut is the sum of edges that crosses the cut (go from L to R).
- A (*s*, *t*)-cut partitions graph such that L contains *s* and all vertices *s* can reach and R contain *t* and all vertices that can reach *t*.
- For any flow *f* and any (*s*, *t*)-cut (L, R), size(*f*) ≤ capacity(L, R).
- The capacity of the cuts provide an upper bound of the maximum flow.
- Size of maximum flow equals to capacity of smallest *s*-*t* cut.

MINIMUM CUT

- {S} and {A, C, B, D, T} forms a min cut.
- {S, A, B, C, D} and {T} forms another min cut.
- Minimum cut will contain only edges in the residual graph that goes from R to L.
- In the original graph, only edges from L to R in the minimum cut.
- These edges are fully saturated (at full capacity).



REDUCTIONS

- When using Max Flow to solve problems, make use of reduction.
- Reduction: solve problem A by solving problem B.
  - Suppose we have a problem A,
  - Preprocess input to fit inputs for problem B (suppose max flow).
  - Solve problem B.
  - Postprocess output of problem B to fit output of problem A.
- Example. Use Max flow to solve network flows with supply and demand nodes.